Introducing Multiple Preference Criteria in Defeasible Argumentation

Nicolás D. Rotstein Guillermo R. Simari Alejandro J. García

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur. Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: ++54 291 4595135 - Fax: ++54 291 4595136
{ndr,grs,ajg}@cs.uns.edu.ar

In this research line we are exploring the possibility of having multiple preference criteria to compare arguments in in Defeasible Argumentation. In this work we are assuming a goal-oriented Argumentative System, to which an application or an agent may seek an answer in response to a query. Thus, based on the current knowledge available, the system will decide whether there exists undefeated support for that query.

1 Introduction

A defeasible argumentation system provides ways to confront contradictory statements in order to determine whether some particular information can be supported. The result of the argumentation process leads to a certain answer, and involves many stages, namely: argument construction, attack, and warrant. In the second one, attack, there exists the need to compare the attacked argument with its attacker to decide which one prevails. In that manner, the definition of a formal criterion for comparing arguments becomes a central problem in defeasible argumentation ([1, 3]).

Given an argument structure $\mathcal{A}_2$ that is a counter-argument for $\mathcal{A}_1$, in order to decide which one prevails, these two arguments must be compared by some criterion. For example, in [2], if the counter-argument $\mathcal{A}_2$ is better than $\mathcal{A}_1$ w.r.t. the comparison criterion used, then $\mathcal{A}_2$ prevails and it will be called a proper defeater for $\mathcal{A}_1$. If neither argument is better nor worse than the other, a blocking situation occurs, and we will say that $\mathcal{A}_2$ is a blocking defeater for $\mathcal{A}_1$. If $\mathcal{A}_1$ is better than $\mathcal{A}_2$, then $\mathcal{A}_2$ will not be considered as a defeater for $\mathcal{A}_1$, and $\mathcal{A}_1$ prevails.

Since the argument comparison criterion is a fundamental part of argumentation systems, its implementation is directly related to the kind of inferences that will be made by an agent. If the criterion is inappropriate, blocking situations will often arise.

The advantage of having an argumentative system with a single comparison criterion, is that the encoder of the knowledge has to keep in mind just this single policy, and it will be

Partially supported by SGCyT Universidad Nacional del Sur, CONICET and Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096 and PICT 2003 Nro 15043)

clear when an argument should prevail. If the preference criterion is based on some property that must be given explicitly, such rule preference, then the knowledge representation could be heavily biased by that criterion, turning this characteristic into a disadvantage.

Nevertheless, not having the possibility of making dynamic changes over the comparison criterion bounds the application or agent that performs the queries to simple domains, or to very similar domains where the criterion maintains its usefulness. It is difficult to conceive a "real" agent with social abilities and a single preference criterion. Usually, the criteria used to compare arguments changes according to the interaction domain. An agent attached to a single criterion often will be frustrated when the interaction domain changes.

2 Multiple Comparison Criteria

The possibility of having multiple criteria could improve a system with many components as well as an agent that faces different situations. The analysis that follows covers both.

First consider a system that has many components, each one of them has to perform inferences from knowledge represented in different domains. These systems need to have multiple comparison criteria available. Otherwise, they will have to face many blocking situations. Another interesting case is that of an agent that has to face different situations, with very different restrictions. It wouldn't be able to achieve this without multiple criteria. A good example of this may be a soccer-playing agent that has to change the way that it considers the possibility of performing a certain action (*e.g.*, shoot to score a goal), according to the game's current result. Further considerations regarding this example will be presented in Section 3.

2.1 Supporting Multiple Criteria

There are some features that have to be built into an argumentative system in order to support multiple criteria. The following is a description of the components that this kind of defeasible argumentation system should include:

- A preference criteria set (CS). This set will contain every criterion that may prevail on a given moment.
- A way to create (possibly ordered) subsets CS_i out of CS. This is useful for the agent to be prepared to face rather different situations, associating predetermined subsets of preference criteria to different queries. This subsets have to be dynamic in the sense that environmental and specific situations could affect agent's preferences, and the dynamism of the world should be reflected on the set.
- Primitives to modify those sets (i.e., CS and CS_i), in order to add and/or remove any of its elements. This is required for providing the above mentioned dynamism.

- A mechanism to establish an order among the criteria that belong to any subset of *CS*. This ordering is important when an agent needs to perform a query and several criteria can be used. When a criterion is not enough, then it is clear the next criterion to use.

- Means to dynamically associate criteria subsets with queries (see Figure 1). Having this feature allows an argumentation system to automatically switch criteria subsets when a query is performed. This implies that the agent won't have to explicitly set the domain when it performs a query. It must be also possible to change these associations (the agent may "change his mind").

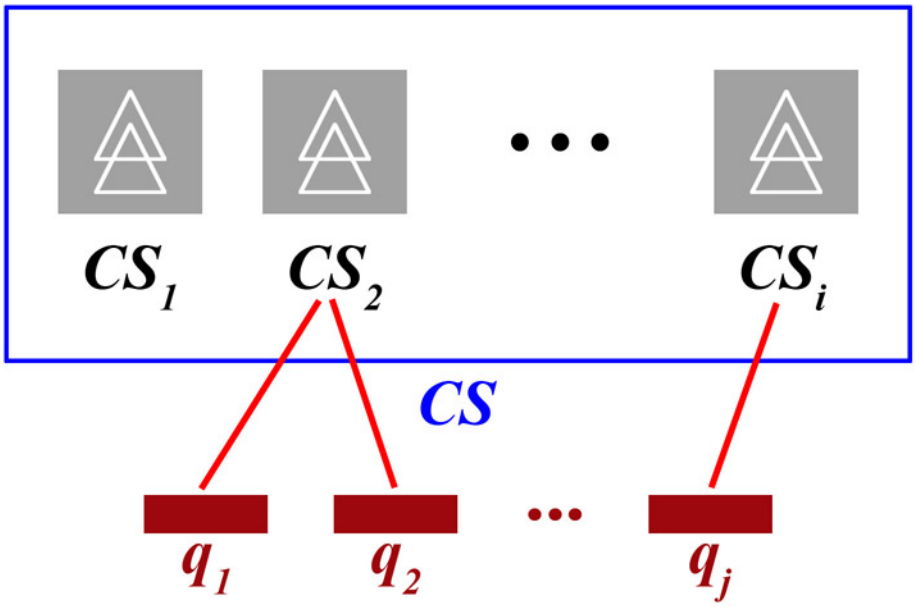

Figure 1: *Interaction between criteria subsets and queries*

- The ability of obtaining the answer corresponding to each criterion from the CS_i associated with a query.

2.2 Advantages of this Approach

Many advantages arise in an argumentative system with multiple preference criteria. Some of them are the following:

- Is more general. Usually, the use of a single comparison criterion implies that, in many cases, the conflicting arguments result incomparable. Considering this, it is sensible to think that, when that situation arises, a different criterion could be used.

- Versatility. Being able to manage the subsets of *CS* at will, according to the current application, makes the system more versatile. This means that any of the changes in agent's preferences may have its correlation in the criteria it will use.

- Solves the problem of having a set of queries implicitly associated to a set of criteria. In order to achieve this, the programmer has to establish those links and the system itself will react to a query switching to the corresponding CS_i. Being application-dependent, setting up the links connecting queries and criteria is up to the programmer.

- The search space of a problem can be reduced by successive queries. Consider an application that tries to find a solution to a problem. To find that solution, the program queries its knowledge regarding some general property of the problem and obtains an answer (possibly indecisive). According to this, it will continue querying in a more specific way until the solution is found.

- Credulous and skeptical conclusions could be inferred from a given knowledge base. An application can query the available knowledge using all of the criteria on a CS_i in order to infer a skeptical conclusion if all of the queries (with all the different criteria) were warranted. On the other hand, a credulous conclusion will be inferred if at least one of the queries was warranted.

- The ability of obtaining multiple answers for a query (corresponding to each criterion from the associated CS_i) could be used by an agent to dynamically reorder, or filter, the CS_i set.

3 Example: soccer-playing agent

In this section we will describe how multiple criteria is an important feature to extend a defeasible argumentation system. It will also help clarify some ideas stated above.

Robotic soccer has proven to be a complex enough system to test many of the features of any reasoning system. The robots are controlled by software agents, each of which has a set of high-level actions to perform, such as kicking the ball with a given strength or moving in a given direction. Every moment an agent has to choose which action to do next, and that choice can be made by using a reasoning system, in this case, a defeasible argumentation system. The result of the argumentation process is highly dependent on the argument comparison policy, since it solves all of the conflicts (*i.e.*, attacks) between arguments. Therefore, every change in the preference criteria implies that the answer of a given query may vary.

Usually a (human) soccer player reacts to a change in the result of the game by modifying his preferences. In certain moments, he has to determine, for instance, whether to shoot to try to score a goal or not, and he does so by taking into consideration the state of the game: if his team is losing, he will shoot more often.

Considering the design of a software agent, this personality changes could be achieved through the use of multiple comparison criteria. Whenever the state of game varies, the agent switches to the CS_i that better fits the current situation, by performing an ad-hoc query. When a new situation appears, a new CS_i could be built and its corresponding link to a query must

be established by the programmer. During execution, the detection of that particular situation will trigger the query that switches to the recently created CS_i.

The soccer-playing agent will have a CS_i for every game state that he can recognize. For instance, he may change his attitude only by taking into account the current result of the game. Therefore, his subsets of CS will be: CS_{lose}, CS_{tie} and CS_{win}.

The elements of this subsets refer to preference criteria regarding to: shooting on goal, passing the ball to a teammate, marking a certain opponent and carrying the ball, respectively. The resulting interaction between queries and comparison criteria, as well as how the action is performed can be observed in Figure 2.

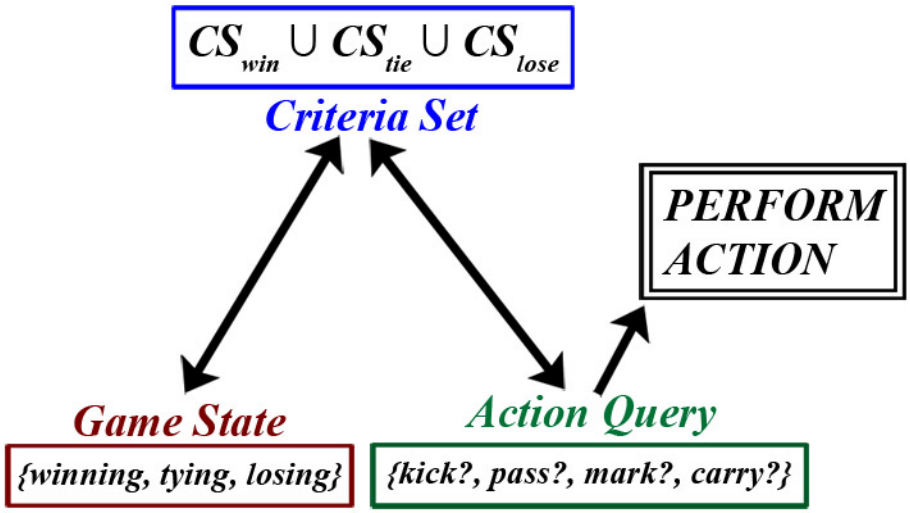

Figure 2: *Criteria subsets, queries and actions for a soccer-playing agent*

Finally, when game state changes, the agents reflects this change by performing the corresponding query, which is associated to an appropiate CS_i.

References

[1] Carlos I. Chesñevar, Ana G. Maguitman, and Ronald P. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.

[2] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 2003. Accepted for publication. http://xxx.lanl.gov/abs/cs.AI/0302029.

[3] Henry Prakken and Gerard Vreeswijk. Logical systems for defeasible argumentation. In D.Gabbay, editor, *Handbook of Philosophical Logic, 2nd ed.* Kluwer Academic Pub., 2000.